



JCTVC-K0204: VPS syntax for scalable and 3D extensions

Jill Boyce

Vidyo



Introduction

- **High level syntax for VPS extensions proposed, changes relative to JCTVC-J1007**
 1. Simplification of the VPS extension to restrict same number of scalability attribute dimensions for all layers
 2. Renaming of num_hrd_parameters to num_hrd_operation_points
 3. Operation point signaling using per layer flags
 4. Profile & level signaling for operation points
 5. Profile & level signaling in SPS for enhancement layers
- **Some of the proposed changes (#2, #3) impact the base HEVC specification VPS, others impact VPS extension**

1. Scalability attribute dimensions in VPS extension

- In the first approach to layer_id partitioning described in JCTVC-J1007 section 2.2.1
 - num_dimensions_minus1[i] syntax element is sent for each layer i
 - dimension_type[i][j] syntax element are sent in a two-dimensional loop, for each layer i and for each dimension j
 - There is an error in the syntax table for the j loop, which refers to num_dimensions_minus1 instead of num_dimensions_minus1[i]
- Propose to simplify the syntax by sending num_dimensions_minus1 only once in the VPS extension and apply its value to all layers
 - A single dimensional loop over each layer i can then be used for dimension_type[i] syntax element
 - If a scalability attribute will be sent for at least one layer, it seems clearer to require that a value for that attribute be sent for all layers, so that its value is clear for all layers
 - Change may increase VPS extension bit count for case when different number of attributes are sent for different layers, but reduces bit count in typical case when number of attributes is same for all layers

1. Proposed Syntax for scalability attribute dimensions in VPS extension

vps_extension() {	Descriptor
while(!byte_aligned())	
vps_extension_byte_alignment_reserved_zero_bit	u(1)
for(i = 1; i <= vps_max_layers_minus1; i++) {	
// mapping of layer ID to scalability dimension IDs	
num_dimensions_minus1[i]	u(4)
for(j = 0; j <= num_dimensions_minus1; j++) {	
dimension_type[i][j]	u(4)
dimension_id[i][j]	u(8)
}	
num_dimensions_minus1	u(4)
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_type[j]	u(4)
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j <= num_dimensions_minus1; j++)	
dimension_id[i][j]	u(8)
// layer dependency	
num_direct_ref_layers[i]	u(6)
for(j = 0; j < num_direct_ref_layers[i]; j++)	
ref_layer_id[i][j]	u(6)
}	
}	

2. Renaming of num_hrd_parameters to num_hrd_operation_points

- **Operation points descriptions sent in the initial section of the VPS (prior to the extension flag) for the HRD, may be re-used in the VPS extension**
 - Impacts HEVC base specification, but only as an editorial issue
- **Re-naming of the syntax element clarifies that operation point descriptors, and not just HRD parameters are (optionally) present in the VPS**

2. Proposed syntax for renaming of num_hrd_parameters to num_hrd_operation_points

video_parameter_set_rbsp() {	Descriptor
video_parameter_set_id	u(4)
vps_temporal_id_nesting_flag	u(1)
reserved_zero_2bits	u(2)
max_num_layers_minus1 //reserved_zero_6bits	u(6)
vps_max_sub_layers_minus1	u(3)
profile_level(1, vps_max_sub_layers_minus1)	
next_essential_info_byte_offset //reserved_zero_12bits	u(12)
for(i = 0; i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase[i]	ue(v)
}	
num_hrd_operation_points num_hrd_parameters	ue(v)
for(i = 0; i < num_hrd_operation_points num_hrd_parameters ; i++) {	
if(i > 0)	
operation_point(i)	
hrd_parameters(i == 0, vps_max_sub_layers_minus1)	
}	
...	

3. Operation point signaling using per layer flags

- **Proposed change impacts the HEVC base specification**
- **Revised signaling of operation point layer signaling is proposed**
 - Simpler
 - Uses only fixed length coding
 - Requires fewer syntax elements
 - Uses fewer bits in typical usage where relatively few layers are present in the coded bitstream, but does not restrict flexibility

3. Proposed syntax for operation point signaling using per layer flags

operation_point(opIdx) {	Descriptor
op_num_layer_id_values_minus1 [opIdx]	ue(v)
for(i = 0; i <= op_num_layer_id_values_minus1; i++)	
op_layer_id [opIdx][i]	u(6)
for(i = 0; i <= max_num_layers_minus1; i++)	
layer_present_in_op_flag[opIdx][i]	u(1)
}	

layer_present_in_op_flag[opIdx][i] equal to 1 specifies that layer i is present in operation point opIdx, equal to 0 specifies that layer i is not present in operation point opIdx.

3. Proposed syntax for operation point signaling using per layer flags

- Proposed change requires that the anticipated use of the **reserved_zero_6bits** in the VPS for **max_num_layers_minus1** be included in the base specification
 - operation_point() syntax refers to it

video_parameter_set_rbsp() {	Descriptor
video_parameter_set_id	u(4)
vps_temporal_id_nesting_flag	u(1)
reserved_zero_2bits	u(2)
max_num_layers_minus1 //reserved_zero_6bits	u(6)
vps_max_sub_layers_minus1	u(3)

4. Profile & level signaling for operation points

- **Proposed syntax for profile & level signaling for each operation point in the VPS extension**
- **Re-use the HRD operation points from the initial section of the VPS**
- **Additional operation points may be sent in the VPS extension**
- **Profile and level indicators optionally sent for all operation points**
- **For each operation point, send a `profile_present_flag`**
 - When equal to 0, instead of sending the profile related syntax elements for that operation point, a `profile_op_ref` syntax element is sent to indicate a reference profile from a previously sent operation point

4. Proposed syntax for profile & level signaling for operation points

Code	Descriptor
<code>vps_extension() {</code>	
<code>while(!byte_aligned())</code>	
<code>vps_extension_byte_alignment_reserved_zero_bit</code>	<code>u(1)</code>
<code>// layer specific information</code>	
<code>for(i = 1; i <= vps_max_layers_minus1; i++) {</code>	
<code>// mapping of layer ID to scalability dimension IDs</code>	
<code>num_dimensions_minus1[i]</code>	<code>u(4)</code>
<code>for(j = 0; j <= num_dimensions_minus1; j++) {</code>	
<code>dimension_type[i][j]</code>	<code>u(4)</code>
<code>dimension_id[i][j]</code>	<code>u(8)</code>
<code>}</code>	
<code>// layer dependency</code>	
<code>num_direct_ref_layers[i]</code>	<code>u(6)</code>
<code>for(j = 0; j < num_direct_ref_layers[i]; j++)</code>	
<code>ref_layer_id[i][j]</code>	<code>u(6)</code>
<code>}</code>	
<code>// op specific information</code>	
<code>num_additional_operation_points</code>	<code>u(8)</code>
<code>for(i = 0 ; i < num_additional_operation_points; i++)</code>	
<code>operation_point(i + num_hrd_operation_points)</code>	
<code>for(i = 1; i <= num_hrd_operation_points +</code>	
<code>num_additional_operation_points; i++) {</code>	
<code>vps_profile_present_flag[i]</code>	<code>u(1)</code>
<code>if (!vps_profile_present_flag[i])</code>	
<code>profile_op_ref[i]</code>	<code>u(8)</code>
<code>profile_tier_level(vps_profile_present_flag[i],</code>	
<code>vps_max_sub_layers_minus1)</code>	
<code>}</code>	
<code>}</code>	

num_additional_operation_points specifies the maximum number of additional operation points present in the coded video sequences the video parameter set applies.

vps_profile_present_flag[i] equal to 1 specifies the profile information for operation point i is present in the `profile_tier_level()` syntax structure. `vps_profile_present_flag[i]` equal to 0 specifies that profile information for operation point i is not present in the `profile_tier_level()` syntax structure. When `vps_profile_present_flag[i]` equal to 0, profile information for operation point i is inferred to be equal to the profile information of operation point **profile_op_ref[i]**.

5. Profile & level signaling in SPS for enhancement layers

- **In base HEVC specification, VPS can be considered optional for the base layer, and hence the profile & level information is duplicated in the SPS**
- **For enhancement layers the VPS is required**
 - No need to replicate the profile information signaling in the SPS for enhancement layers
 - Add condition to send profile information in SPS only when layer_id is equal to 0
 - Still send level information in the SPS for enhancement layers, to allow changing of level for an individual layer without requiring an IDR be sent for all layers
 - Interpret level information in the SPS for enhancement layers as applying only to the individual layer
 - Differs from operation point level signaling in the VPS which corresponds to cumulative layers

5. Proposed syntax for profile & level signaling in SPS for enhancement layers

- **Proposed constraint definition improves coding efficiency vs. the MVC and SVC methods**
 - Reducing the number of SPSs required to be transmitted.
 - Reduced bits for the signaling of the PPS identifier in the coded slice header, because fewer different SPS id values must be signaled, and hence fewer different PPS id values
 - PPS value in the slice header is variable length entropy coded, such that higher values require more bits than smaller values

	Descriptor
seq_parameter_set_rbsp() {	
video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_reserved_zero_bit	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
profile_tier_level(layer_id == 0 , sps_max_sub_layers_minus1)	
seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)

Conclusion

- **5 proposed modifications to VPS**

1. Simplification of the VPS extension to restrict same number of scalability attribute dimensions for all layers
2. Renaming of num_hrd_parameters to num_hrd_operation_points
3. Operation point signaling using per layer flags
4. Profile & level signaling for operation points
5. Profile & level signaling in SPS for enhancement layers

- **#2 and #3 impact base specification**